Using Anticipative Malware Analysis to Support Decision Making

Mathieu Couture and Frédéric Massicotte
Communications Research Centre Canada
3701 Carling Avenue
Ottawa, Ontario
CANADA

{Mathieu.Couture, Frederic.Massicotte}@crc.gc.ca

ABSTRACT

A software tool allowing the safe monitoring of the execution of malicious software (malware), or more generally, programs that cannot be trusted is commonly referred to as a sandbox. *Most of the time, a sandbox is implemented in a virtual machine or a simulated operating system and allows the behaviour of the program to be studied from the host's point of view. We are investigating the usefulness of a sandbox in the context of decision making. More specifically, we have designed and implemented a* network sandbox, *i.e. a sandbox that allows us to study malware behaviour from the network perspective. We plan to use this sandbox to generate malware-sample profiles that can be used by decision making algorithms to help network administrators and security officers decide on a course of action to be followed upon detection of a malware threat. This paper focuses on the implementation details of the sandbox. It is flexible enough to allow the study of malware behaviour in the presence of any given configuration of software and operating system. It also allows the user to specify the network topology to be used.*

1 INTRODUCTION

Once the presence of a malicious software (malware) threat has been detected on a computer, the questions that arise immediately include: Is this threat directly targeted at us? What is the presence of the malware threat implying about our security posture? What damage has it potentially caused? What should we do in order to minimize the additional damage it may cause? The answer to the last question may involve several steps such as disconnecting the infected computer from the network, disinfecting the infected computer, scanning the other computers to verify whether the threat has propagated, etc. Two factors will determine which of those actions should be undertaken and in which order: the nature of the threat, and the specific contextual role that the infected computer plays in the network. For example, a malware that has been delivered through an e-mail apparently coming from a friend or a colleague should be treated differently than a malware that has been caught through web browsing (what was the infection vector?). Similarly, an end-user computer may not necessitate the same care as a central database server (what are the resources that are affected?). Finally, a widely known botnet that is generally used to perform click fraud may not be treated with the same urgency as a distributed denial of service attack that seems to be directly targeted (what is the primary purpose of the threat?).

We are currently collaborating in a research project on computer network defence decision making and contributing to the objective to improve decision making capabilities by addressing existing deficiencies with information analysis tools. We are developing an anticipative approach to malware analysis to support military commanders, network operators and cyber-defence analysts in decision making.

The anticipative approach relies on the pre-existence of a database of malware-sample behaviour. It contrasts with the traditional reactive approach, where one has to capture the executable code before identifying which specific malware is causing the attack. We foresee two main advantages with using the anticipative approach. First, the response time can be significantly reduced. Second, it is not required to capture the binary as the database can be queried as soon as attack symptoms are detected.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
NOV 2010	**N/A**	**-**

4. TITLE AND SUBTITLE
Using Anticipative Malware Analysis to Support Decision Making

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Communications Research Centre Canada 3701 Carling Avenue Ottawa, Ontario CANADA

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADA564697. Information Assurance and Cyber Defence (Assurance de l'information et cyberdefense). RTO-MP-IST-091

14. ABSTRACT
A software tool allowing the safe monitoring of the execution of malicious software (malware), or more generally, programs that cannot be trusted is commonly referred to as a sandbox. Most of the time, a sandbox is implemented in a virtual machine or a simulated operating system and allows the behaviour of the program to be studied from the host's point of view. We are investigating the usefulness of a sandbox in the context of decision making. More specifically, we have designed and implemented a network sandbox, i.e. a sandbox that allows us to study malware behaviour from the network perspective. We plan to use this sandbox to generate malware-sample profiles that can be used by decision making algorithms to help network administrators and security officers decide on a course of action to be followed upon detection of a malware threat. This paper focuses on the implementation details of the sandbox. It is flexible enough to allow the study of malware behaviour in the presence of any given configuration of software and operating system. It also allows the user to specify the network topology to be used.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**10**	

Our research plan consists in three phases:

1. designing and implementing tools allowing continual anticipative analysis of malware samples;
2. designing and implementing algorithms that summarize the raw information generated by the above tools into qualitative descriptions which can be used by the decision making algorithms; and
3. designing and implementing decision making algorithms that use the above summaries together with contextual information to provide advice about the course of action to be followed.

At this point, we have done most of phase 1, i.e. we have designed and implemented a software tool that generates raw information about malware samples. This software tool is the main topic of this paper. We call it the Automatic Experimentation System (AES). Although the AES was primarily designed to provide us with dynamic information (i.e. derived from observations made during malware execution), it is flexible enough to also provide us with static information such as anti-virus scan reports. We are currently using it for both purposes. However, in this paper, the focus is on how we use it to obtain dynamic information.

The rest of this paper is organized as follows: in Section 2, we present an overview of related work on malware analysis. In Section 3, we present past projects that led to the development of the AES. In Section 4, we present the system architecture of the AES. In Section 5, through a typical use-case example, we present the XML scheme that is used to configure the AES. In Section 6, we discuss observations made after the execution of 30000 malware samples within the AES. We conclude in Section 7.

2 RELATED WORK

Over time, different approaches have been adopted to analyse malware samples, each falling into one of the two following categories: the dynamic approach (where the code is being executed) and the static approach (where the code is being analysed without being executed). Each category has its pros and cons, but an important advantage of the dynamic approach is that it reveals subtleties that could not be identified by static analysis either because the executable is obfuscated using techniques such as self-extracting code, or because it exploits some unknown vulnerability of the operating system, or just because its structure is too complex. On the other hand, dynamic analysis has the inconvenience of presenting a security risk since it involves the execution of software that is designed to perform undesirable actions.

The efforts that have been deployed to circumvent the last problem led to the establishment of a research approach known as Sandboxing. A sandbox is a software tool that allows the safe monitoring of the execution of malware or other kinds of programs. From a functional point of view, a sandbox is a process where the input is an executable file and the output is a set of reports about the different actions taken by the program during the monitoring period. These reports may contain information about modifications made to the file system or the registry, the network activity, the information displayed on the screen, or anything else considered relevant by the designer. A report can consist of organized textual information or binary files such as network traffic capture files.

Well-known sandboxes include Norman Sandbox [1], CWSandbox [2], and Anubis [3]. The approach of Norman Sandbox is to execute the program in a simulated operating system, while the two other sandboxes execute the program in the real operating system inside a virtual machine, i.e. only the hardware part is simulated. To monitor the actions taken by the program, CWSandbox uses a technique called *API hooking*, which consists in replacing the system DLLs of Windows by home-made DLLs that log all the system calls being made before forwarding them to the real DLLs. Anubis, on the other hand, monitors system calls and API functions from outside the virtual machine. This technique has the advantage of leaving Windows unmodified. In addition, Anubis allows the monitored virtual machine to

connect to the Internet [4]. Recording traffic traces has the advantage of providing the analyst with an execution trace of the malware from a networking point of view. To avoid causing damage, some filtering is done about which traffic is actually routed on the Internet. Even if there are clear advantages as to what can be learned by monitoring malware executing in a network connected to the Internet, there are certainly some downsides, as this activity must be performed with extreme caution in order to avoid spreading the infection over the Internet.

The AES allows the generation of traffic traces resulting from the execution of malware in an environment that is completely isolated from the Internet. The purpose of these traces is to characterize malware behaviour from a network point of view. The AES supports the execution of malware in a customisable virtual network that aims to emulate a corporate network connected to the Internet. As for CWSandbox and Anubis, the operating systems and the applications that are installed and executed within the virtual machines are *real*, i.e. they behave the same way as if they were installed on a physical machine. In particular, they present all the same vulnerabilities, regardless of whether they have been documented or not. In addition, the AES is flexible enough to integrate any operating system and software combination of choice. When, for a given organization, a typical software configuration is available, this allows a more realistic description of the malware behaviour with respect to that organization.

3 BACKGROUND

In 2005, we started a project whose objective was to produce labelled attack traffic traces in an effort to provide the security community with a new set of traces given that the ones provided by DARPA [5] were already five years old. That led to the development of the AES, which we used initially to generate attack traffic traces capturing the execution of 124 exploits against 108 different target operating systems [6].

The first version of the AES uses desktop computers as the hardware and VMware Workstation as the virtualisation technology. To speed-up the process of generating the traces, we designed a simple synchronization mechanism allowing multiple computers to parallelise their efforts. Over time, the AES used between 25 and 30 computers (typically, having a Pentium 4 processor and 2 GB of RAM) in our laboratory.

Executing malware involves a higher level of risk than executing exploits. In the case of exploits, we generally know prior to execution what is the intended purpose of the software, which is in most cases is a simple proof of concept of how to take advantage of a given vulnerability. In the case of malware, there is much higher risk as the samples arrive undocumented. Our actual goal is to automatically generate the documentation. A possible situation, that has already been observed, is the highjacking of the physical host by the malware.

If that happened, the efforts required to recover from the potential damages of such an attack could be cumbersome. Regardless of the virtualisation technology that is used, such an attack still might occur. Our solution to this problem was to go for a design that would reduce the number of hardware components involved. This way, it would be easier to automate the recovery process in case of an incident. The choice that we made was to use multi-core servers with a high amount of memory instead of desktop computers and to use VMware ESX instead of VMware Workstation.

4 SYSTEM ARCHITECTURE

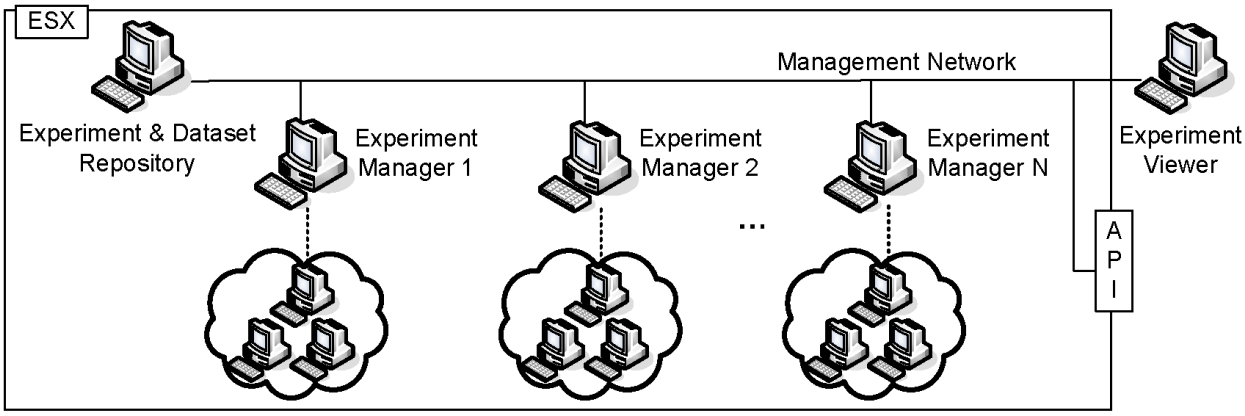

Figure 1: AES Architecture

Figure 1 presents the architecture of our system in the case where a single server is used. When more than one server is used, the architecture extends simply. The system is designed so that several experiments can be performed simultaneously within one server. An experiment consists of a sequence of steps to be performed by a set of virtual machines that are connected to each other according to a specific network topology. In our current setup, we perform ten simultaneous experiments per server. The experiments to be performed are specified in an XML format. That format is described in Section 5. The XML files containing the experiment specifications are stored in a virtual machine called the Experiment & Dataset (E&D) Repository. That machine also stores the malware samples as well as the files that are produced during the execution of the experiments.

Each individual experiment is performed by a virtual machine called Experiment Manager (EM). There are as many EMs as there are simultaneous experiments to be performed. Each EM has its own set of isolated virtual networks and virtual machines (called actors) to perform the experiments. The EMs and the E&D Repository share a common network (called the Management Network) that is used to acquire the list of experiments to be performed and to store the results of the experiments. The ESX server itself is also connected to the Management Network. This connection allows the EMs to interact with the ESX API to control the actors (e.g. starting and stopping the machines, reverting them to an initial snapshot, connecting them to a given virtual network, connecting their CD-ROM drives to a given ISO image, connecting the serial ports of two virtual machines together). Finally, there is one physical machine other than the ESX server that is connected to the Management Network. It is used to visualize the current state of the AES. We call it the Experiment Viewer.

The ESX API does not in itself allow the EMs to run processes on the actors. Since, for obvious security reasons, the actors do not share any network connection with the EMs, a regular SSH connection cannot be used either. Therefore, we had to develop our own secured communication channels (depicted in dotted lines in Figure 2). To start and stop processes on the actors, we developed a custom-made protocol using the serial port. Since the EMs are virtual machines, they can use the ESX API to connect their own serial port to the actors' ports and control the actors from there. However, to transfer relatively large files such as malware-samples (having a size anywhere between a few KB and 10 MB) and network traffic traces (often having a size near 100 MB) the serial port is inappropriate in terms of efficiency. To transfer the malware in the virtual machines, we use a virtual CD-ROM image that we connect to the virtual machine before the experiment execution. To get the traffic traces out of the virtual machines we use WinImage[1] to

[1] www.winimage.com

extract them directly from the virtual machine's hard drive images. To communicate through these custom-made communication channels, we designed a software agent that needs to be installed on the actors (only the ones on which we need to execute processes).

5 USE CASE EXAMPLE

In this section, we describe a specific experiment that we recently performed with the AES. We also use that experiment as an example to give an impression of the expressiveness of our XML format. The objective of the experiment was to see how much information we can obtain by executing malware in an isolated network environment. We plan to perform several iterations of that experiment with more and more sophisticated network architectures, but as this paper is being written, only the first iteration is completed.

```
1.  <experiment executiongraph="S1.S2.S3.S4.S5.S6.S7">
2.  <actor id="VMWin2KSP4Malware">
3.      <nic interface="ethernet0" VMNet="3" />
4.      <cdrom>
5.          <file path="M:/malware/samples/a1b2c3.exe" />
6.      </cdrom>
7.  </actor>
8.  <actor id="VMLinuxRH80DNAT">
9.      <nic interface="ethernet0" VMNet="2" />
10.     <nic interface="ethernet1" VMNet="3" />
11. </actor>
12. <actor id="VMWin2KSP4Target">
13.     <nic interface="ethernet0" VMNet="2" />
14. </actor>
15. <actor id="VMLinuxFC4SnifferVMNet2">
16.     <nic interface="ethernet0" VMNet="2" />
17. </actor>
18. <actor id="VMLinuxFC4SnifferVMNet3">
19.     <nic interface="ethernet0" VMNet="3" />
20. </actor>
21. <actor id="VMLinuxFedora9Honeypot">
22.     <nic interface="ethernet0" VMNet="2" />
23. </actor>
24. <step id="S1" actor="VMLinuxFC4SnifferVMNet2" cmd="START  SNIFFER eth0"  />
25. <step id="S2" actor="VMLinuxFC4SnifferVMNet3" cmd="START  SNIFFER eth1"  />
26. <step id="S3" actor="VMWin2KSP4Malware" cmd="cmd /c D:\a1b2c3.exe" />
27. <step id="S4" actor="MANAGER"  cmd="SCREENSHOT VMWin2KSP4Malware" />
28. <step id="S5" actor="MANAGER" cmd="SCREENSHOT VMWin2KSP4Target" />
29. <step id="S6" actor="VMLinuxFC4SnifferVMNet2" cmd="STOP  SNIFFER eth0"  />
30. <step id="S7" actor="VMLinuxFC4SnifferVMNet3" cmd="STOP  SNIFFER eth1"  />
31. </experiment>
```

Figure 2: XML Experiment File Example

Through our participation in the WOMBAT project [7], we gained access to 30000 malware samples. We executed each malware sample against two different target machines. Therefore, a total of 60000 experiments were performed.

Figure 2 shows a simplified version of the XML file that was used to run the experiment corresponding to a specific malware sample against a specific target. Our XML format has two main parts. The first part (lines 2 to 23) consists of the description of the network topology to be used and some of the actors' configurations. The second part (lines 24 to 30) consists of the description of the sequential steps to be performed during the experiment. Some steps are implicitly performed in every experiment and are thus not specified in the XML file. Such steps include starting and stopping the virtual machines and reverting them to an initial snapshot.

5.1 Network Topology

The experiment that we performed involved six actors and two networks. Figure 3 (lower part) shows the network topology specified by the XML file shown in Figure 2. The actor called VMWin2KSP4Malware is the one on which the malware actually executes. It is connected to the network called VMNet3 and receives the malware through its CD-ROM drive. The execution of the malware generates connections to IP addresses that are a priori unknown. To solve this, we use a Destination Network Address Translator[2]. The actor called VMLinuxRH80DNAT fulfills that role. It routes the traffic coming from VMNet3 to VMNet2 according to the destination port rather than the destination IP address.

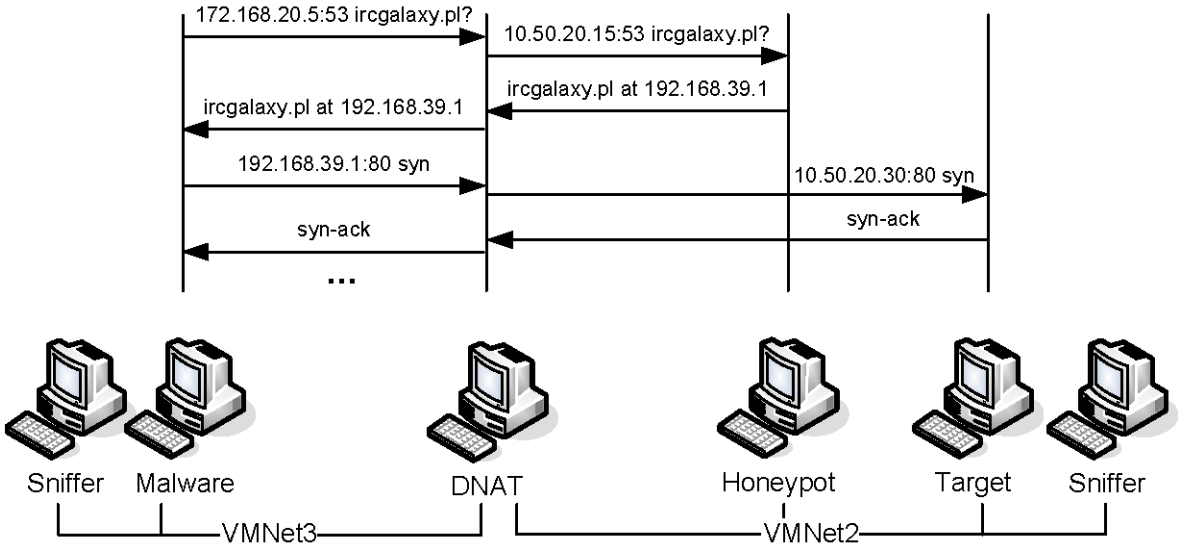

Figure 3: Network Topology and Typical Communication Pattern Involving DNS Traffic

Figure 3 (upper part) shows a sequence diagram of a typical case involving DNS traffic. First, the Malware machine issues a DNS request for ircgalaxy.pl. The request is re-routed by the DNAT to the Honeypot machine, which provides an arbitrary IP address. Then, the Malware machine initiates a connection on port 80 to that address, and the DNAT forwards it to the target machine.

In the case of the script shown on Figure 2, the target machine is VMWin2KSP4Target, a Windows 2000 Professional Service Pack 4 machine. The other machine that we used as a target was a Windows XP Home Edition. The Windows 2000 machine was configured with the default ports opened as well as ports 21, 23 and 80. The Windows XP machine was only configured with the services that are typically available on a home desktop.

[2] linux.die.net/man/8/iptables

The aim of the target machine is to capture the behaviour of a potential propagation mechanism. The malware executes on VMWin2KSP4Malware, and all the traffic it generates to destination addresses outside the range of IP addresses of VMNet3 (except for DNS) is directed to the target machine. In particular, propagation traffic should reach the target machine.

The aim of the VMLinuxFedora9Honeypot machine is to provide or emulate services that might be needed by the malware. In the first iteration of our experiment, the only emulated service was DNS. We wrote a simple script that, for each new DNS request, provides an arbitrary IP address. We observed that numerous malware samples need a DNS in order to go further in their execution. Thanks to our DNAT machine, whichever DNS server the malware intends to reach, our script will handle the request. In future iterations, we will add more emulated (or real if possible) services. For example, an IRC server would be a good component to have.

5.2 Network Topology

The second part of the XML file shown in Figure 2 consists of the list of sequential steps to be performed by the actors (lines 24 to 30). The order in which the actions must be executed is specified in the executiongraph property of the experiment (line 1). We plan to support parallel actions in a future version of our system if it becomes required.

The sequence of actions to be performed in the case of the experiment depicted in Figure 2 is straightforward: start the sniffers (lines 24 and 25), execute the malware (line 26), take screenshots (lines 27 and 28), and stop the sniffers (lines 29 and 30). Due to space constraints, some details have been omitted.

6 DISCUSSION AND FUTURE WORK

Analysis of the traffic traces generated using the AES in the setup described in Section 5 provided us with useful information about the 30000 malware samples such as the TCP and UDP ports on which they are trying to establish connections, the different IP addresses they are trying to contact, the various DNS names that they are trying to resolve, etc. Analysing the traffic traces with various intrusion detection systems could provide us with information about the various IDS alarms one should expect to see when the malware samples are being executed in a real network.

6.1 Comparing with Anti-Virus Vendors Reports

An interesting observation to be made is that since we also scanned the samples using various anti-virus scanners, we could counter-verify the reports provided by the vendors on their web sites. Often, our observations were similar but we have also observed slight deviations from the documented behaviours, which are probably due to collisions in the anti-virus signatures. To offer better protection, anti-virus vendors tend to design signatures that are as generic as possible while maintaining a low false positive rate. A problem with this strategy is that the documentation is written upon analysis of a single sample (or at least, not as many samples as the signature will eventually match). Therefore, documenting per sample is still desirable, although problems brought in by polymorphic worms render the population of an absolute database containing all samples ever observed impossible. An alternative may be to give lower priority to polymorphic worms for which a great number of instances have already been analysed.

6.2 Improvements to be Made

We could have gathered more information if we had provided the AES with a better network environment. For example, we neglected to include an SMTP server accepting connections on port~25. Having

configured an SMTP server accepting all e-mails sent by the various malware samples, we could also have gathered information about e-mail recipients, attachments, the text of the e-mails, etc. Also, we neglected to include an IRC server. If we had done this, we could have gathered information about chat rooms, nick names, text sent to the channel, etc. An interesting aspect of IRC is that it can be observed on various different ports. Therefore, it is not possible, prior to malware-samples execution, to decide on which port the IRC daemon should be listening. It would be nice to rely on the DNAT to dynamically translate the port numbers appropriately, but as far as we know, it is not possible to configure iptables to do this. Similarly, we observed that some malware samples attempt to open a port on the target machine. When the attempt is not successful and the malware sample tries to connect on that port, the target machine simply sends a reset and all information that we could have gained by listening to that connection is then lost. Again, a dynamic port forwarding strategy would have been helpful.

6.3 Decision Making

As stated in Section 1, one of the reasons why we built the AES is that we believe that the information it generates can be useful in the context of decision making. Our hypothesis is that the information that can be automatically generated using the AES could be stored into a malware database that, if properly designed, could be consulted by decision makers and/or automated tools in the context of decision making.

We believe that it is possible to design a decision support system that can search the database and correlate the observed behaviour of a live malware threat with the ones stored in the database. One of the numerous challenges of the anticipative approach is to make sure that the system is designed so that it is not misled by meaningless similarities. Nevertheless, we believe that a database of malware-sample behaviour can be useful to decision support systems in the same way that experience is useful to human beings.

In the following two years, our focus will be on the decisions that have to be made upon detection of a malware threat in a military context. To achieve this goal, the questions to which we are seeking answers include the following:

1) Can this capability support decision making at different levels (e.g. military commanders versus network operators)?
2) What kind of decisions do decision makers have to make?
3) Do decision makers need a system that makes recommendations or do they simply need information that is meaningfully summarised?
4) How would decision makers expect to interact with such a system?
5) What is the level of abstraction that is expected by decision makers?

The answers to these questions will help us design a database and a database interface that are relevant for the purpose of decision making. Our implementation of the AES has shown that it will be feasible to populate the database.

7 FINAL COMMENTS

The experiments that we performed demonstrated that the AES is useful to automate the analysis of malware samples. Most of the required improvements pointed out in the preceding section do not concern the AES itself but the way it is used, for example the configuration of the virtual machines that are used during a given experiment.

Analysis in an isolated environment, completely disconnected from the Internet, is a design choice that we made to avoid causing any harm. Given proper filtering mechanisms, it would be technically feasible to

use the AES with a live Internet connection. This would certainly lead to the generation of more informative profiles. However, we do not believe that this is necessary to demonstrate the usefulness of the AES.

We used virtualisation to facilitate mass analysis. However, this strategy presents the inconvenience of being detectable. Several malware samples are equipped with virtual machine detection capabilities that allow them to determine whether they are executing on a physical or on a virtual machine. To avoid analysis, some of these malware samples will simply abort their execution or present a misleading behaviour when they detect that they are executing on a virtual machine. To design a hypervisor that is not detectable by malware represents an important challenge that is beyond the context of our research.

ACKNOWLEDGEMENTS

This work was supported by Defence Research and Development Canada.

REFERENCES

[1] Norman Solutions: Norman sandbox whitepaper. http://download.norman.no/whitepapers/whitepaper_Norman_SandBox.pdf (2003)

[2] Willems, C., Holz, T., Freiling, F.: Toward automated dynamic malware analysis using cwsandbox. IEEE Security and Privacy 5(2) (2007) 32-39

[3] Bayer, U., Moser, A., Krügel, C., Kirda, E.: Dynamic analysis of malicious code. Journal in Computer Virology 2(1) (2006) 67-77

[4] Bayer, U., Milani Comparetti, P., Hlauscheck, C., Kruegel, C., Kirda, E.: Scalable, Behavior-Based Malware Clustering. In: 16th Symposium on Network and Distributed System Security (NDSS). (2009)

[5] Lincoln Laboratory Massachusetts Institute of Technology: Darpa intrusion detection evaluation. http://www.ll.mit.edu/IST/ideval/data/data_index.html

[6] Massicotte, F., Gagnon, F., Labiche, Y., Briand, L.C., Couture, M.: Automatic evaluation of intrusion detection systems. In: ACSAC, IEEE Computer Society (2006) 361-370

[7] Leita, C., Pham, V.H., Thonnard, O., Ramirez Silva, E., Pouget, F., Kirda, E., Dacier, M.: The leurre.com project: collecting internet threats information using a worldwide distributed honeynet. In: 1st WOMBAT workshop, April 21st-22nd, Amsterdam, The Netherlands. (Apr 2008)